EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	30	michael near marr.in.	US-PGPUB; USPAT	OR	ON	2007/07/30 08:44
S2	4	scott near brender.in.	US-PGPUB; USPAT	OR	ON	2007/07/30 08:44
S3	17513	microsoft.as.	US-PGPUB; USPAT	OR	ON	2007/07/30 08:45
S4	120	S3 and (call$4 and method and functionality).clm.	US-PGPUB; USPAT	OR	ON	2007/07/30 08:46
S5	1	S4 and (source and stack).clm.	US-PGPUB; USPAT	OR	ON	2007/07/30 08:46
S6	21	S3 and (stack and call and invok$4).clm.	US-PGPUB; USPAT	OR	ON	2007/07/30 08:57
S7	19	("5592600"|"5794047"|"5802371"|"5892900"|"5948113"|"5970248"|"6173421"|"6240549"|"6513155"|"6662358"|"20030041267"|"20020188931"|"20020012432"|"20020013772"|"20020169974"|"20030187801"|"20030194092"|"20030195855"|"20030226007").PN.	US-PGPUB; USPAT	OR	ON	2007/07/30 11:17
S8	50	("5826250" "5949424" "5880736" "6163319" "6327694" "20050231605" "6289507" "6823517" "20050073575" "4296391" "4894846" "5432934" "5668928" "6021440" "6044216" "6240460" "6349000" "6416389" "6434529" "6519562" "6587939" "20040006465" "4951292" "4344143" "5782946" "3905699" "3881100" "3816729" RE29864 "4599703" "4607281" "4934814" "5056144" "5260897" "5291497" "5384722" "5387826" "5408104" "5444640" "5479078" "5619702" "5623545" "5625730" "5640590" "5727029" "5734720" "5745625" "5745881" "5758257" "5761248"). pn.	US-PGPUB; USPAT	OR	ON	2007/07/30 11:18
S9	3270	717/124-135.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/30 11:28

EAST Search History

S10	168	S9 and (call and stack and (return adj address))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/30 11:30
S11	159	S10 and (@pd<"20040301" or @ad<"20040301" or @prad<"20040301" or @rlad<"20040301")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/30 11:36
S12	45	S11 and dll	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/30 11:31
S13	2520	(return adj address) same stack	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/30 11:36
S14	130	S13 and dll and call$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/30 11:36
S15	120	S14 and (@pd<"20040301" or @ad<"20040301" or @prad<"20040301" or @rlad<"20040301")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/30 11:36
S16	30	("4542453" | "4701847" | "5093916" | "5113369" | "5155847" | "5247681" | "5274819" | "5297282" | "5303378" | "5375241" | "5394545" | "5410698").PN. OR ("5734904").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2007/07/30 11:52
S17	42	("5247678" | "5247681" | "5339430" | "5369766" | "5369770" | "5375241" | "5450586" | "5454086" | "5475840").PN. OR ("5946486").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2007/07/30 12:05

EAST Search History

S18	303	modif$5 with (previous near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:25
S19	19	S18 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:12
S20	635	717/110-113.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:25
S21	19	S20 and (previous near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:30
S22	506	(modified or modification) with (previous$3 near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:35
S23	24	S22 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:31
S24	8	("3670310" | "5185885" | "5287447" | "5355497" | "5367671" | "5473772" | "5602993" | "5768566").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2007/07/31 15:33
S26	11	(US-20020188931-$).did. or (US-5970248-$ or US-5802371-$ or US-5794047-$ or US-5946486-$ or US-6003095-$ or US-6708330-$ or US-6698015-$ or US-5734904-$ or US-6442752-$ or US-5410698-$). did.	US-PGPUB; USPAT	OR	ON	2007/07/31 15:36

EAST Search History

S27	9	S26 and state	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:36
S28	1470	(detect$4 or check$4) with ((software or program) near3 modif$5)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:38
S29	147	S28 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:38
S30	19	("5592600"|"5794047"|"5802371"|"5892900"|"5948113"|"5970248"|"6173421"|"6240549"|"6513155"|"6662358"|"20030041267"|"20020188931"|"20020012432"|"20020013772"|"20020169974"|"20030187801"|"20030194092"|"20030195855"|"20030226007").PN.	US-PGPUB; USPAT	OR	ON	2007/07/31 15:46
S31	17	S30 and state	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:47
S32	1	S30 and (previous$4 near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:50
S33	16118	decrypt$4 near key	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:51
S34	11562	decrypt$4 adj key	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:51

EAST Search History

S35	584	cryptographic$4 with (decryption adj key)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 15:52
S36	241	S35 and tamper$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:08
S37	84	S35 and tamper$4resistant	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:09
S39	0	(software or program) with "not" with modifi$4 with (previous adj state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:21
S40	2	(software or program) with modifi$4 with (previous adj state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:21
S41	584	(sofware or program) with (previous near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:24
S42	42	S41 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:27
S43	19	S20 and (previous near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:30

EAST Search History

S44	5	(software or program) with modified with (previous near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:30
S48	692	(modified or alter$4 or changed) with (previous near state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:38
S49	18	S48 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:36
S54	5029	encrypt$4 near3 (decryption adj key)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:45
S55	41	S54 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:45
S56	33	("4222068" | "4521852" | "4521853" | "4613901" | "4634808" | "4696034" | "4887296" | "5029207" | "5237610" | "5293424" | "5436621" | "5442704" | "5592552" | "5619247" | "5754647" | "5757919" | "5764762" | "5774546" | "5799080" | "5802274" | "5809140" | "5825878" | "5852290" | "5892899" | "5892900" | "5923759" | "5982899" | "5999623" | "6009177" | "6041412" | "6044155" | "6049608" | "6069957").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2007/07/31 16:48
S57	4	software with hid$4 with (decryption adj key)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:50

EAST Search History

S58	562	tamper$5resistant and (decryption adj key)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:51
S59	115	S58 and dll	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 16:51
S60	237	"717".clas. and (modified near3 state)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:00
S62	2	"6738970".pn.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:03
S63	10	("5872979" | "6038393" | "6052531" | "6131192" | "6167567" | "6199198" | "6216140" | "6216175" | "6226652" | "6226747").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2007/07/31 17:05
S64	744	(detect$4 or check$4) with ((software or program) near3 modification)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:19
S65	67	S64 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:19
S66	1470	(detect$4 or check$4) with ((software or program) near3 modif$5)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:19

EAST Search History

S67	147	S66 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:20
S68	104	S67 and state	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:20
S69	941	checksum with modif$5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:22
S70	23	S69 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/31 17:22

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

call stack verification dll

USPTO

Feedback Report a problem Satisfaction survey

Terms used: **call stack verification dll**

Found **34,329** of **207,474**

Sort results by	relevance
Display results	expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200
Best 200 shown

Result page: **1** 2 3 4 5 6 7 8 9 10 next

Relevance scale

1 Workshop on architectural support for security and anti-virus (WASSA): Energy-security tradeoff in a secure cache architecture against buffer overflow attacks
Koji Inoue
March 2005 **ACM SIGARCH Computer Architecture News**, Volume 33 Issue 1
Publisher: ACM Press
Full text available: pdf(391.20 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we propose a cache architecture, called *SCache*, to detect buffer-overflow attacks at run time. Furthermore, the energy-security efficiency of SCache is discussed. SCache generates replica cache lines on each return-address store, and compares the original value loaded from the memory stack to the replica one on the corresponding return-address load. The number and the placement policy of the replica line strongly affect both energy and vulnerability. In our evaluation, it i ...

2 Cryptography and data security
Dorothy Elizabeth Robling Denning
January 1982 Book
Publisher: Addison-Wesley Longman Publishing Co., Inc.
Full text available: pdf(19.47 MB) Additional Information: full citation, abstract, references, cited by, index terms

From the Preface (See Front Matter for full Preface)

Electronic computers have evolved from exiguous experimental enterprises in the 1940s to prolific practical data processing systems in the 1980s. As we have come to rely on these systems to process and store data, we have also come to wonder about their ability to protect valuable data.

Data security is the science and study of methods of protecting data in computer and communication systems from unauthorized disclosure ...

3 Defensive technology: Detection of injected, dynamically generated, and obfuscated malicious code
Jesse C. Rabek, Roger I. Khazan, Scott M. Lewandowski, Robert K. Cunningham
October 2003 **Proceedings of the 2003 ACM workshop on Rapid malcode WORM '03**
Publisher: ACM Press
Full text available: pdf(240.68 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents DOME, a host-based technique for detecting several general classes of malicious code in software executables. DOME uses static analysis to identify the locations (virtual addresses) of system calls within the software executables, and then monitors the executables at runtime to verify that every observed system call is made from a location identified using static analysis. The power of this technique is that it is simple, practical, applicable to real-world software, and high ...

Keywords: anomaly detection, code analysis, dynamic analysis, execution monitoring, intrusion detection, malicious code detection, static analysis, system calls

4 Applications: Repairing return address stack for buffer overflow protection
Yong-Joon Park, Gyungho Lee
April 2004 **Proceedings of the 1st conference on Computing frontiers CF '04**
Publisher: ACM Press
Full text available: pdf(197.90 KB) Additional Information: full citation, abstract, references, citings, index terms

Although many defense mechanisms against buffer overflow attacks have been proposed, buffer overflow vulnerability in software is still one of the most prevalent vulnerabilities exploited. This paper proposes a micro-architecture based defense mechanism against buffer overflow attacks. As buffer overflow attack leads to a compromised return address, our approach is to provide a software transparent micro-architectural support for return address integrity checking. By keeping an uncompromised cop ...

Keywords: buffer overflow, computer architecture, computer security, intrusion tolerance

5 Workshop on architectural support for security and anti-virus (WASSA): Using DISE to protect return addresses from attack
Marc L. Corliss, E. Christopher Lewis, Amir Roth
March 2005 **ACM SIGARCH Computer Architecture News**, Volume 33 Issue 1
Publisher: ACM Press
Full text available: pdf(389.57 KB) Additional Information: full citation, abstract, references, index terms

Stack-smashing by buffer overflow is a common tactic used by viruses and worms to crash or hijack systems. Exploiting a bounds-unchecked copy into a stack buffer, an attacker can---by supplying a specially-crafted and unexpectedly long input---overwrite a stored return address and trigger the execution of code of her choosing. In this paper, we propose to protect code from this common form of attack using dynamic instruction stream editing (DISE), a previously proposed hardware mechanism that im ...

6 SPiKE: engineering malware analysis tools using unobtrusive binary-instrumentation
Amit Vasudevan, Ramesh Yerraballi
January 2006 **Proceedings of the 29th Australasian Computer Science Conference - Volume 48 ACSC '06**
Publisher: Australian Computer Society, Inc.
Full text available: pdf(832.66 KB) Additional Information: full citation, abstract, references, index terms

Malware -- a generic term that encompasses viruses, trojans, spywares and other intrusive code -- is widespread today. Malware analysis is a multi-step process providing insight into malware structure and functionality, facilitating the development of an antidote. Behavior monitoring, an important step in the analysis process, is used to observe malware interaction with respect to the system and is achieved by employing dynamic coarse-grained binary-instrumentation on the target system. However, ...

Keywords: instrumentation, malware, security

7 Security: A framework for trusted instruction execution via basic block signature verification

Milena Milenković, Aleksandar Milenković, Emil Jovanov
April 2004 **Proceedings of the 42nd annual Southeast regional conference ACM-SE 42**
Publisher: ACM Press
Full text available: pdf(276.25 KB) Additional Information: full citation, abstract, references, citings

Most of today's computers are connected to the Internet or at least to a local network, exposing system vulnerabilities to the potential attackers. One of the attackers' goals is the execution of the unauthorized code. In this paper we propose a framework that will allow execution of the trusted code only and prevent malicious code from executing. The proposed framework relies on the run-time verification of basic block signatures. The basic block signatures are generated during a trusted instal ...

Keywords: computer security, intrusion detection, trusted execution

8 Pluggable verification modules: an extensible protection mechanism for the JVM
Philip W. L. Fong
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10
Publisher: ACM Press
Full text available: pdf(224.39 KB) Additional Information: full citation, abstract, references, index terms

Through the design and implementation of a JVM that supports Pluggable Verification Modules (PVMs), the idea of an extensible protection mechanism is entertained. Link-time bytecode verification becomes a pluggable service that can be readily replaced, reconfigured and.augmented. Application-specific verification services can be safely introduced into the dynamic linking process of the JVM. This feature is enabled by the adoption of a previously proposed modular verification architecture, Pro ...

Keywords: Aegis VM, Java virtual machine, bytecode verification, extensible protection mechanism, extensible systems, mobile code security, pluggable verification modules, proof linking

9 Session 2:'Review and analysis of synthetic diversity for breaking monocultures
James E. Just, Mark Cornwell
October 2004 **Proceedings of the 2004 ACM workshop on Rapid malcode WORM '04**
Publisher: ACM Press
Full text available: pdf(356.14 KB) Additional Information: full citation, abstract, references, citings, index terms

The increasing monoculture in operating systems and key applications and the enormous expense of N-version programming for custom applications mean that lack of diversity is a fundamental barrier to achieving survivability even for high value systems that can afford hot spares. This monoculture makes flash worms possible. Our analysis of vulnerabilities and exploits identifies key assumptions required to develop successful attacks. We review the literature on synthetic diversity techniques, f ...

Keywords: diversity, n-version programming, vulnerability

10 Going native: Module-aware translation for real-life desktop applications
Jianhui Li, Peng Zhang, Orna Etzion
June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments VEE '05**
Publisher: ACM Press
Full text available: pdf(584.15 KB) Additional Information: full citation, abstract, references, index terms

A dynamic binary translator is a just-in-time compiler that translates source architecture binaries into target architecture binaries on the fly. It enables the fast running of the source architecture binaries on the target architecture. Traditional dynamic binary translators invalidate their translations when a module is unloaded, so later re-loading of the same module will lead to a full retranslation. Moreover, most of the loading and unloading are performed on a few "hot" modules, which caus ...

Keywords: dynamic binary translation, dynamic loaded module, memory management, translation reuse

11 Security: Hardware support for code integrity in embedded processors
Milena Milenković, Aleksandar Milenković, Emil Jovanov
September 2005 **Proceedings of the 2005 international conference on Compilers, architectures and synthesis for embedded systems CASES '05**
Publisher: ACM Press
Full text available: pdf(371.76 KB) Additional Information: full citation, abstract, references, index terms

Computer security becomes increasingly important with continual growth of the number of interconnected computing platforms. Moreover, as capabilities of embedded processors increase, the applications running on these systems also grow in size and complexity, and so does the number of security vulnerabilities. Attacks that impair code integrity by injecting and executing malicious code are one of the major security issues. This problem can be addressed at different levels, from more secure softwa ...

Keywords: attacks, code injection, code integrity

12 The equivalence problem for real-time DPDAs
Michio Oyamaguchi
July 1987 **Journal of the ACM (JACM)**, Volume 34 Issue 3
Publisher: ACM Press
Full text available: pdf(2.51 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The equivalence problem for deterministic real-time pushdown automata is shown to be decidable. This result is obtained by showing that Valiant's parallel stacking technique using a replacement function introduced in this paper succeeds for deterministic real-time pushdown automata. Equivalence is also decidable for two deterministic pushdown automata, one of which is real-time.

13 Vigilante: end-to-end containment of internet worms
Manuel Costa, Jon Crowcroft, Miguel Castro, Antony Rowstron, Lidong Zhou, Lintao Zhang, Paul Barham
October 2005 **ACM SIGOPS Operating Systems Review , Proceedings of the twentieth ACM symposium on Operating systems principles SOSP '05**, Volume 39 Issue 5
Publisher: ACM Press
Full text available: pdf(329.29 KB) Additional Information: full citation, abstract, references, citings, index terms

Worm containment must be automatic because worms can spread too fast for humans to respond. Recent work has proposed network-level techniques to automate worm containment; these techniques have limitations because there is no information about the vulnerabilities exploited by worms at the network level. We propose Vigilante, a new end-to-end approach to contain worms automatically that addresses these limitations. Vigilante relies on collaborative worm detection at end hosts, but does not requir ...

Keywords: control flow analysis, data flow analysis, self-certifying alerts, worm containment

14 Sealing OS processes to improve dependability and safety

Galen Hunt, Mark Aiken, Manuel Fähndrich, Chris Hawblitzel, Orion Hodson, James Larus, Steven Levi, Bjarne Steensgaard, David Tarditi, Ted Wobber
March 2007 **ACM SIGOPS Operating Systems Review , Proceedings of the 2007 conference on EuroSys EuroSys '07**, Volume 41 Issue 3
Publisher: ACM Press
Full text available: pdf(281.05 KB) Additional Information: full citation, abstract, references, index terms

In most modern operating systems, a process is a hardware-protected abstraction for isolating code and data. This protection, however, is selective. Many common mechanisms---dynamic code loading, run-time code generation, shared memory, and intrusive system APIs---make the barrier between processes very permeable. This paper argues that this traditional *open process architecture* exacerbates the dependability and security weaknesses of modern systems.

As a remedy, this paper prop ...

Keywords: open process architecture, sealed kernel, sealed process architecture, software isolated process (SIP)

15 Type-Safe linking with recursive DLLs and shared libraries
Dominic Duggan
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Publisher: ACM Press
Full text available: pdf(658.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Component-based programming is an increasingly prevalent theme in software development, motivating the need for expressive and safe module interconnection languages. Dynamic linking is an important requirement for module interconnection languages, as exemplified by dynamic link libraries (DLLs) and class loaders in operating systems and Java, respectively. A semantics is given for a type-safe module interconnection language that supports shared libraries and dynamic linking, as well as circular ...

Keywords: Dynamic Linking, Module Interconnection Languages, Recursive Modules, Shared Libraries

16 Mobile code: Anomaly intrusion detection in dynamic execution environments
Hajime Inoue, Stephanie Forrest
September 2002 **Proceedings of the 2002 workshop on New security paradigms NSPW '02**
Publisher: ACM Press
Full text available: pdf(867.25 KB) Additional Information: full citation, abstract, references, index terms

We describe an anomaly intrusion-detection system for platforms that incorporate dynamic compilation and profiling. We call this approach "dynamic sandboxing." By gathering information about applications' behavior usually unavailable to other anomaly intrusion-detection systems, dynamic sandboxing is able to detect anomalies at the application layer. We show our implementation in a Java Virtual Machine is both effective and efficient at stopping a backdoor and a virus, and has a low false positi ...

Keywords: Java, anomaly detection, dynamic sandboxing

17 Selected writings on computing: a personal perspective
Edsger W. Dijkstra
January 1982 Book
Publisher: Springer-Verlag New York, Inc.

Additional Information: full citation, abstract, references, cited by, index terms

Since the summer of 1973, when I became a Burroughs Research Fellow, my life has been very different from what it had been before. The daily routine changed: instead of going to the University each day, where I used to spend most of my time in the company of others, I now went there only one day a week and was most of the time that is, when not travelling!-- alone in my study. In my solitude, mail and the written word in general became more and more important. The circumstance that my employe ...

18 A formal framework for component deployment
Yu David Liu, Scott F. Smith
October 2006 **ACM SIGPLAN Notices , Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications OOPSLA '06**, Volume 41 Issue 10
Publisher: ACM Press
Full text available: pdf(592.54 KB) Additional Information: full citation, abstract, references, index terms

Software deployment is a complex process, and industrial-strength frameworks such as .NET, Java, and CORBA all provide explicit support for component deployment. However, these frameworks are not built around fundamental principles as much as they are engineering efforts closely tied to particulars of the respective systems. Here we aim to elucidate the fundamental principles of software deployment, in a platform-independent manner. Issues that need to be addressed include deployment unit design ...

Keywords: component, deployment, version

19 Workshop on architectural support for security and anti-virus (WASSA): Using instruction block signatures to counter code injection attacks
Milena Milenković, Aleksandar Milenković, Emil Jovanov
March 2005 **ACM SIGARCH Computer Architecture News**, Volume 33 Issue 1
Publisher: ACM Press
Full text available: pdf(283.67 KB) Additional Information: full citation, abstract, references, citings, index terms

With more computing platforms connected to the Internet each day, computer system security has become a critical issue. One of the major security problems is execution of malicious injected code. In this paper we propose new processor extensions that allow execution of trusted instructions only. The proposed extensions verify instruction block signatures in run-time. Signatures are generated during a trusted installation process, using a multiple input signature register (MISR), and stored in an ...

20 Thread-modular shape analysis
Alexey Gotsman, Josh Berdine, Byron Cook, Mooly Sagiv
June 2007 **ACM SIGPLAN Notices , Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation PLDI '07**, Volume 42 Issue 6
Publisher: ACM Press
Full text available: pdf(298.58 KB) Additional Information: full citation, abstract, references, index terms

We present the first shape analysis for multithreaded programs that avoids the explicit enumeration of execution-interleavings. Our approach is to automatically infer a resource invariant associated with each lock that describes the part of the heap protected by the lock. This allows us to use a sequential shape analysis on each thread. We show that resource invariants of a certain class can be characterized as least fixed points and computed via repeated applications of shape analysis only o ...

Keywords: abstract interpretation, concurrent programming, shape analysis, static analysis

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next